The distributed p-median problem in computer networks

The Third International Workshop on Parallel and Distributed Data Mining of the 19th International Conference on Computational Science and Its Applications (ICCSA 2019), Saint Petersburg, Russia, July 1-4 2019. [in press]

The Distributed p-Median Problem in Computer Networks

Anas AlDabbagh

Department of Computer Science, University of Mosul, Iraq

Giuseppe Di Fatta

Department of Computer Science, University of Reading, UK

Antonio Liotta

Department of Electronics, Computing and Mathematics, University of Derby, UK

Abstract

Many distributed services in computer networks rely on a set of active facilities that are selected among a potentially large number of candidates. The active facilities then contribute and cooperate to deliver a specific service to the users of the distributed system. In this scenario graph partitioning or clustering is often adopted to determine the most efficient locations of the facilities. The identification of the optimal set of facility locations is known as the p-median problem in networks, is NP-hard and is typically solved by using heuristic methods. The goal is to select p locations among all candidate network nodes such that some cost function is minimised. A typical example of such a function is the overall communication cost to deliver the service to the users of the distributed system. Locating facilities in near-optimal locations has been extensively studied for different application domains. Most of these studies have investigated sequential algorithms and centralised approaches. However, centralised approaches are practically infeasible in large-scale and dynamic networks, where the problem is inherently distributed or because of the large communication overhead and memory requirements for gathering complete information about the network topology and the users. In this work distributed approaches to the p-median problem are investigated. Two solutions are proposed for addressing the facility locations problem in a fully distributed environment. Two different iterative heuristic approaches are applied to gradually improve a random initial solution and to converge to a final solution with a local minimum of the overall cost. While the first approach adopts a fine granularity by identifying a single change to improve the solution at each iteration, the second approach applies changes to every component of the solution at each iteration. An experimental comparative analysis based on simulations has shown that the approach with a finer granularity is able to deliver a better optimisation of the overall cost with longer convergence time. Both approaches have excellent scalability and provide an effective tool to optimise the facility locations from within the network. No prior knowledge of the system is required, no data needs to be gathered in a centralised server and the same process is used to identify and to deploy the facility locations solution in the network since the process is fully decentralised.

1 Introduction

Locating facilities in near-optimal locations to deliver some service to users has received a significant interest [1] [2] [3]. A facility is an object or node (such as server or a network device) that provides services such as a distributed memory cache, while users are the network nodes that benefit from the service offered by these facilities. A facility can be either in an open or closed state: an open facility means it can serve connected clients to it, while a closed one is a candidate location where a facility can be opened if required. To provide a more efficient service to the users the topological location of an open facility should be close to many user nodes. The p-median problem, in particular, aims to identify the set of the open facilities (medians) among all the candidate locations such that the overall cost of the solution is minimized according to an objective function [3][4][5]. In many scenarios the cost of the solution is associated to the communication overhead, e.g. the total number of hops in the shortest paths that connect each user node to its closest open facility. This formulation is very similar to the classic partitional clustering problem in data mining, best represented by algorithms such as k-means and k-medoids, where the goal is to find the ideal locations of k centres that minimise the sum of the squared distances from each item to its closest centre.

The main motivation for this work is that previous works on the p-median problem have used a centralised approach based on heuristic methods for finding near-optimal solutions. In that case the required information

needs to be collected in order to apply a sequential algorithm to find a solution. A centralised approach is infeasible in large-scale networks due to the time and space complexity of the sequential algorithms as well as the large communication cost and latency to aggregate the global information [6, 7, 8]. Therefore, this work investigates distributed algorithms to solve the p-median problem directly within distributed environments.

In this paper, two new approaches for solving the p-median problem in a distributed environment are proposed. Both are designed to be executed without any centralised collection of the data in a single node. They apply an iterative heuristic approach to improve a random initial solution and to converge to a final solution with a local minimum of the cost.

The first approach builds a global view of the system and improves a current solution by replacing a single facility at each iteration. The second approach, is designed on the logic of the k-medoids clustering algorithm. At each iteration, a local view of each cluster is generated and all facilities can be updated to optimise the solution.

Both approaches were implemented in *PeerSim*, a Java-based network simulator, for investigating the performance in large-scale systems with several tests with different parameters such as the network size, the number of facilities to be located, the total number of candidate facilities and different initial states. The results have shown that the first protocol is more accurate at selecting the locations of the facilities, since it converges to a lower total cost of the solution than the second protocol. However, the second one has shown a better convergence time in optimising the solution.

The rest of the paper is organised as follows. Section 2 introduces the problem, the notation and discusses some related work. Sections 3 and 4 describe the two proposed methods. Section 5 presents the simulations and the comparative experimental results. Finally, Section 6 provides conclusive remarks.

2 The p-Median Problem Definition, Formulation and Related Work

The p-median problem intends to find p locations of active (open) facilities among several candidate locations in a way that the total distance (cost) from the user nodes to the open facility nodes is minimised according to an objective function [9] [10] [11]. As most location problems, the p-median problem is classified as NP-hard and solved using heuristic methods [12].

The p-median problem formulation [3] [4] is formally defined as follows. Let us consider a graph $G = (V, E)$, where V is the set of nodes and E the set of links between nodes, the set $F = \{f_1, f_2, ..., f_m\}$ of m facilities, the set $U = \{u_1, u_2, ..., u_n\}$ of n users, where $V = F \cup U$, an integer number $p < m$, which is the target number of open facilities, and a distance function $d : U * F \rightarrow \mathbb{N}$ of the number of hops between users and facilities. A greedy optimisation procedure is applied to build a solution from an initial set of p randomly selected facilities from F. The procedure seeks to identify a better neighbour solution to the current one. A neighbour solution is a minimal alteration of the current solution that improves the cost.

A vertex substitution procedure developed by Teitz and Bart [4] is one of the standard algorithms for solving the p-median problem [13] [14]. In vertex substitution, p candidate facilities from F are arbitrarily selected (opened) to start the algorithm. The algorithm reallocates (swap) an open facility with one of the candidate (closed) facilities whenever the swap improves the solution. The algorithm then iterates from the existing solution looking for another pair of facilities to be swapped to improve the solution until the best solution is reached. The algorithm is then terminated with a local optimum solution.

Building up on the vertex substitution algorithm, the *fast interchange heuristics* algorithm is proposed by Whitaker [15]. It implements a swap once a profitable facilities pair is found. The removed facility is deleted from F and never comes back to the solution [3].

Another study has been done by Hansen and Mladennovic, who used the *best improvement strategy* in which all possible swaps are evaluated, then the most profitable one is executed. In addition, the way to finding the best facility to open, which is used in the previous method, is evaluated to be less complicated [16].

Based on the above studies, another method to solve the p-median problem was suggested by Resende and Werneck [3]. In their study, all possible swaps were evaluated according to equation 1 and the most profitable pair was chosen for the swap.

$$profit(f_i, f_r) = gain(f_i) - loss(f_r) + extra(f_i, f_r) \tag{1}$$

As noticed from equation 1, the first component *gain* is associated with the candidate facility f_i to be inserted in the solution. The *gain* value is the amount of distance saved by reallocating some user nodes to f_i, for which f_i is closer than their current closest facility. The functions $d_1(u)$ and $d_2(u)$ are used to indicate the distance from the user node u and, respectively, the closest and second closest open facilities $\phi_1(u)$ and $\phi_2(u)$. The *gain* value is computed for all candidates f_i according to the equation 2, so that it results in a vector of size $m - p$.

$$gain(f_i) = \sum_{u \in U} \max\left(0, d_1(u) - d(u, f_i)\right) \tag{2}$$

As a consequence of removing the open facility f_r from the solution, users assigned to it must be reassigned to their second closest open facility ϕ_2, which is further away than f_r by definition. This leads to an additional cost that is identified by a *loss* function and is computed according to the equation 3.

$$loss(f_r) = \sum_{u \in U : \phi_1(u) = f_r} [d_2(u) - d_1(u)] \tag{3}$$

However, some of the user nodes, which were assigned to the removed facility f_r, are in fact assigned to f_i rather than their second closest facility, leading to an additional saving in the total cost. This affects the total cost of the solution and a correction factor called *extra* profit has to be included for these cases. The *extra* value is computed as in equation 4.

$$extra(f_i, f_r) = \sum_{u \in U : [\phi_1(u) = f_r] \wedge [d(u,f_i) < d_2(u)]} [d_2(u) - \max\left(d(u, f_i), d_1(u)\right)] \tag{4}$$

In a distributed environment the information about the network topology, the users and the facilities needs first to be collected in a server in order to compute a solution with a centralised approach. Moreover, once the solution is computed, it will have to be distributed among the nodes in the network for its deployment. This process will then repeat every time there is need to update the solution. In the remainder, two novel distributed protocols, DPM and KM, are presented to solve the p-median problem directly in a distributed environment without prior knowledge about the network topology. No data needs to be gathered in a centralised server and the same process is used to identify and to deploy a p-median solution in the network since the process is intrinsically distributed.

3 The Distributed p-Median (DPM) Protocol

As shown in figure 1, the DPM protocol design is based on three main phases to overcome the synchronisation problems of the distributed environment. It is assumed that an initial solution is available, e.g. an initial random solution of p open facilities. The protocol is started with phase 1 by sending BROADCAST messages from all facilities to all nodes in the network. The user nodes are gradually building a view of facilities in the network and determine the closest open facility. Each user node joins its closest open facility via a JOIN message. From the JOIN messages each open facility builds a view of the user nodes assigned to it.

Thereafter, the facility nodes transit to phase 2. In this phase, the open facilities exchange the required information to build a distributed global view about the network and determine the best pair of facilities to swap in the current solution.

In phase 3, the open facilities implement the swap. Both of phases 2 and 3 are repeatedly executed to improve the solution until convergence.

The following subsections describe the details of each phase of the protocol.

3.1 Phase 1: Initialisation and Information Collection

In Phase 1 all the candidate (open and closed) facilities disseminate a BROADCAST message to advertise their presence and to facilitate the computation of the shortest paths. This message holds specific information about the facility: ID, status and the distance to reach the facility in number of hops. As described in algorithm 1, from the broadcast messages each user node builds a view about the facilities in the network, which is used to select the closest open facility to join, as further explained in the subsections below.

3.1.1 Build a user partial view.

At the start of this phase, the user nodes are kept waiting until receipt of the facility broadcast messages. As soon as a user node receives a new broadcast message, it creates a corresponding *local user record* by extracting the information from the broadcast message payload, such as the ID of the facility that sends the message, the shortest distance to the facility, the status of the facility and some routing information (e.g., the next hop to reach the facility).

Due to the randomness of the flooding approach for the broadcast messages, the user node may receive a broadcast message from the same facility several times via different paths. If the message comes from a shorter path, the user node updates its *local user record*, increments the distance field, and forward the message to all its neighbours except the source.

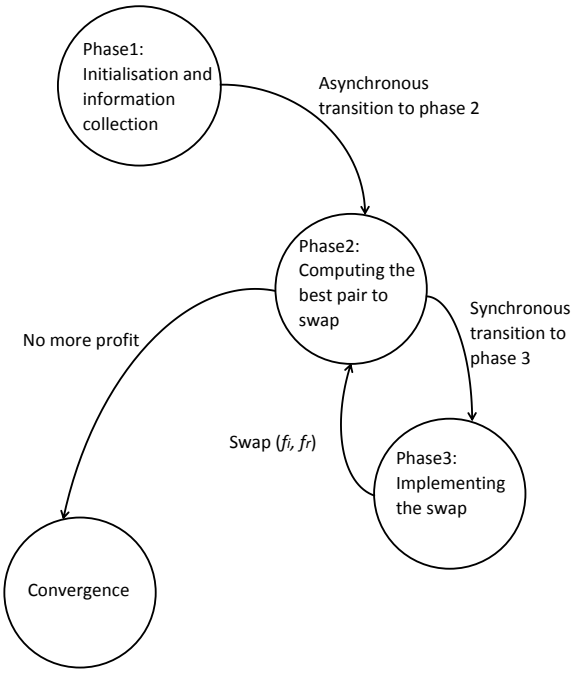


Figure 1: The DPM protocol consists of three main phases. In phase 1 facility information is disseminated over the network and user nodes join their closest open facilities. In phase 2 the open facilities find the best swap of facilities $< f_i, f_r >$ to minimise the cost. In phase 3 the facilities implement the swap.

3.1.2 Join the closest open facility.

While there are no more new messages, the user node computes its loss value, gain and extra vectors. It then sends them to the closest open facility which is determined from its *local user record*.

When the facility receives the join message from a user node, it updates its local record by adding the user's loss value to the local facility value and the user's gain and extra vector values to their corresponding values in the local facility record, as shown in figure (2). When all join messages are received, the open facilities can transit to phase 2.

3.2 Phase 2: Exchange the Necessary Information and Find the Best Pair to Swap

Each open facility f_r is associated to a subset (a cluster) of user nodes, which have identified f_r as the closest open facility from which to receive the service.

At the transition to phase 2, an open facility f_r has determined its local value of loss(f_r) and the two vectors $gain(f_i)$ and $extra(f_i, f_r)$ for the local cluster. In this phase, the open facilities need to exchange and aggregate these local values and vectors to build a global view of the network and of the current p-median solution. Figure 3 shows the collective communication operations (gather and sum), for the aggregation of local information into global information by means of facility *exchange* messages.

The global loss, gain and extra are initialised with the local values and vectors. When an open facility receives an exchange message containing the remote contribution to the global information, it aggregates the loss of the message into a global loss, adds the $gain(f_i)$ to the global gain vector and collects the $extra(f_i, f_r)$ to build a global $extra(f_i, f_r)$.

As shown in algorithm 2, when an open facility receives $p - 1$ number of exchange messages, it computes the profit values according to the equation 1 for all possible pairs $< f_i, f_r >$, where f_i is a closed facility to be inserted in the solution and f_r is an open facility to be removed from the solution. The pair of facilities that provides the greatest profit is chosen for the swap. Based on the shared global view all the open facilities take the same decision by deterministically choosing the same pair of facilities $< f_i^*, f_r^* >$ for the swap and thus transiting to phase 3 to implement the swap.

Algorithm 1: Phase 1 - initialisation and information collection, all the facilities send BROADCAST messages $< FID, distance, status >$ to all nodes, where FID is a unique facility ID and the binary *status* indicates if the facility is open or closed in the current solution. The user nodes build summarised view about the facilities in the network.

```
forall open and closed facilities do
    Send a facility broadcast message m =< FID, 1, status > to all neighbours

At event: a facility broadcast message is received at a user node:
if the m.FID is not in the user local table then
    Add the message payload to the user local table
    m.distance++
    Forward the message to all neighbours except the source
else if the m.distance field < the current matching record.distance then
    Update the local table
    m.distance++
    Forward the message to all neighbours except the source
else
    Drop the message /* The FID message is coming from a longer path */

At event: a facility broadcast message is received at a facility node:
if the m.FID != the local facility ID then
    m.distance++
    Forward the message to all neighbours except the source
else
    Drop the message /* message of the same facility */
```

Algorithm 2: Phase 2 - The open facilities exchange the local information and build the same global view. The best pair of candidate facilities $< f_i^*, f_r^* >$ are selected for a swap.

```
Initialise the global table with the local facility record
Set counter to one
max_profit = 0
f_r^* = null
f_i^* = null
forall open facilities excluding itself do
    send a facility exchange message with the local information records to a remote open facility

At event: receiving a facility exchange message
Add the message payload to the global table
counter++
if (counter == p) then
    for all f_i in the global table do
        for all f_r in the global table do
            compute profit(f_i, f_r)
            if profit(f_i, f_r) > max_profit then
                f_r^* = f_r
                f_i^* = f_i
```

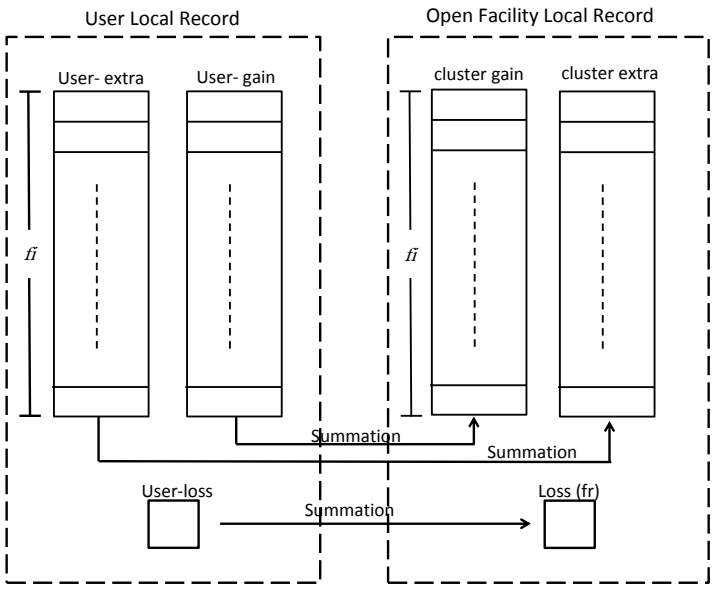


Figure 2: The open facility builds a within-cluster knowledge from the JOIN messages of the user nodes

3.3 Phase 3: Swap Implementation

As shown in algorithm 3, in this phase the candidate f_r^* sends a CHANGE_STATUS message to f_i^* asking to change its status to open and to accept join messages from user nodes. The f_r^* also send a SWAP $< f_i^*, f_r^* >$ message to its user nodes (local cluster). If the user node u is closer to the inserted facility f_i^* than to its second closest open facility $\phi_2(u)$, then u sends a *join* message to f_i^*, otherwise it sends it to $\phi_2(u)$.

The current open facilities in the system inform the user nodes in their clusters about the swap by sending an UPDATE_SOLUTION $< f_i^*, f_r^* >$ message. If a user node u is closer to f_i^* than to its current open facility $\phi_1(u)$, then u leaves its current cluster and sends a join message to f_i^*.

The solution at this stage is reconfigured. Accordingly, the weight values of the clusters are changed. This may lead to find another pair of facilities to swap in the next iteration to improve the solution further. The process is repeated from phase 2 and the total cost keeps improving until reaching the best (local minimum) configuration for the locations of the candidate facilities, given the initial solution.

3.4 Convergence

The convergence status is the final state of the solution, since no more swaps can improve the solution.

4 The k-Medoids (KM) Protocol

The second proposed approach is the k-medoids (KM) protocol. This is inspired by the classic Partitioning Around Medoids (PAM), or k-medoids, clustering algorithm [17]. The main idea of the KM protocol is to partition the network around the open facilities, then to iteratively carry out facility swaps with a similar heuristic method used by the k-medoids clustering algorithm. The KM protocol is organised in three main phases, as explained in the subsections below.

4.1 KM Phase 1: Information Dissemination and Clusters Configuration

Phase 1 is started by the announcement of the facilities about their locations with broadcast messages similarly to the DPM protocol. Nodes gradually build a summarised view of the available open facilities in the network. They build a local view of facilities, including ID, distance and status. Finally user nodes join the closest open facility forming p clusters. A cluster C_k is the set of user nodes associated to the open facility f_k, such that the distance $d(u, f_k)$ is minimal in the current solution, i.e. $\phi_1(u) = f_k$.

During this phase, each open facility (medoid) builds a local table of the closed facilities (candidate medoid) and their associated cost before moving to Phase 2, as shown in algorithm 4.

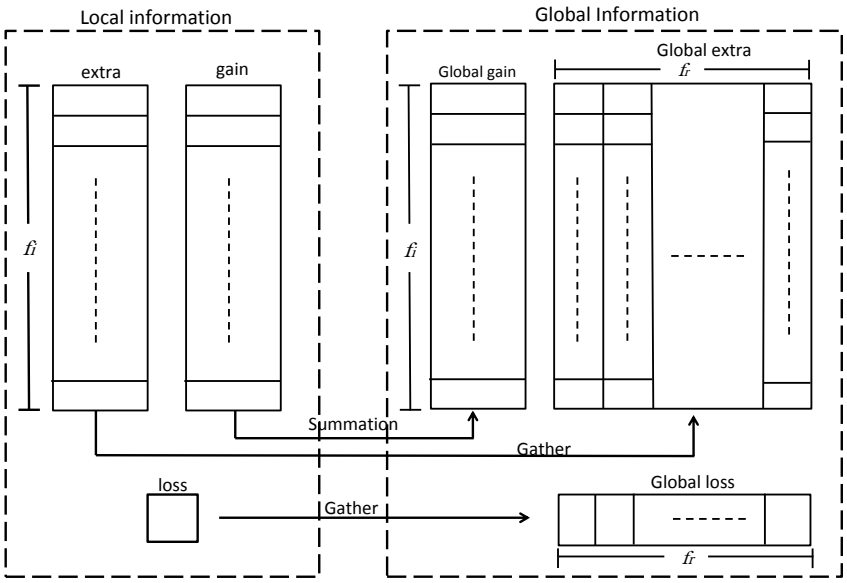


Figure 3: Collective communication operations (gather and sum) among the open facilities for the aggregation of local information to build a global view

4.2 KM Phase 2: Find the Best f_i to Swap

As shown in algorithm 5, each medoid f_k computes the cost of its cluster C_k as the sum of the contributed cost values of the users in the cluster ($u \in C_k$), as per equation (5). Then it evaluates all the candidate facilities that could be opened within its cluster by computing the cost of these closed facilities as if they would be swapped with the current medoid. The total cost of the solution is given in equation (6). If the cost of one of the candidate facility f_i is lower than the cost of the medoid f_k, then the solution is updated in phase 3.

$$cost(C_k) = \sum_{u \in C_k} d(u, f_k) \tag{5}$$

$$cost(\{C_k\}) = \sum_{0<k\leq p} cost(C_k) \tag{6}$$

4.3 KM Phase 3: Update the medoids of the clusters

In this phase, as shown in algorithm 6, all clusters update their medoids independently and in parallel. The update process is implemented as following:

1. The medoid informs the candidate f_i with the swap by sending a CHANGE_STATUS message. At the event of receiving the CHANGE_STATUS message; f_i changes its status to open and get ready for receiving join messages from both user and closed facility nodes.

2. The medoid informs the users and the closed facilities in its cluster of the swap by sending them a SWAP $<f_i, f_r>$ message. The user and the closed facility nodes update their local table by changing the status of f_i to open and the status of f_r to closed, determine a better, if any, closest open facility to join.

3. The medoid informs all other open facilities in the system of the swap by sending CLUSTER_UPDATE$<f_i, f_r>$ message to the medoids of all other clusters. At the event of receiving CLUSTER_UPDATE$<f_i, f_r>$ message, the medoid informs the user and closed facilities nodes in its clusters about the swap.

4.4 Convergence State

Phases 2 and 3 are repeatedly executed and the clusters medoids are updated continuously until no more update messages are received: in this case the facilities make a transition into the convergence state and the algorithm is terminated in a solution with a locally optimal cost.

Algorithm 3: Phase 3 - Implementing the swap $< f_i^*, f_r^* >$

```
if this node is an open facility then
    if this node is f_r^* then
        this.status = closed
        send message CHANGE_STATUS to f_i^*
    forall users in the local cluster do
        Send a swap < f_i^*, f_r^* > message to user
```

```
At event: received message change_status:
this.status = open
```

```
At event: received a swap < f_i^*, f_r^* > message:
Update the local user record
if φ_1 == f_r^* then
    if f_i^* closer than φ_2 then
        send join message to f_i^*
    else
        send join message to φ_2
else if f_i^* closer than φ_1 then
    send leave message to φ_1
    send join message to f_i^*
```

5 Simulation and Experimental Results

A Java-based discrete-event P2P simulation tool called PeerSim [18] is used to simulate the proposed protocols. PeerSim allows simulating large networks with different configurations. In addition to its ability to observe the internal state of nodes in the simulated topology, it allows to keep track of the messages between the facilities and the user nodes. This aids in the performance analysis of the protocols.

The DPM and KM protocols are extensively tested over a range of the network size (up to $500K$ nodes) of artificial and real network topologies.

Figure 4(a) shows the run of both protocols until convergence: the mean and the standard deviation over 10 trials with different randomly chosen initial solutions are shown. The results demonstrate a reduction on the overall cost of the solution at each iteration (swap) until convergence for both protocols. However, KM protocol shows a higher cost of the solution during all cycles. This confirms that the DPM protocol based on global knowledge is more effective in improving the solution cost than the KM protocol, which improves the location of the facilities based on local cluster information.

As shown in figure 4(b), the KM protocol converges faster than the DPM protocol. However, the DPM protocol is found to be capable to identify better final locations for the facilities.

For testing the DPM and KM on different graphs, many trials were carried out on different network sizes (figure 5(a)): the mean cost and the standard deviation of these trials showed for all graph sizes that the DPM protocol provides a lower cost than the KM protocol.

Both protocols are also tested on a varying number (m) of the available candidate locations for the facilities on the same topology. As shown in figure 5(b), DPM takes better advantage than KM of the larger search space for the solution of the p-median problem.

Since both the DPM and KM protocols depend on network messages for addressing the p-median problem, the communication overhead in terms of the number of messages is analysed for the different types of messages. As shown in figure 6, the most significant number of messages among the nodes is the facility BROADCAST messages. These messages flood the network to propagate information from all facilities to all user nodes.

Practically, each facility is meant to serve nodes in its cluster, typically a local part of the topology. Further investigation has shown that it is unnecessary to forward the broadcast messages to the whole network since it causes a heavy load on the network and an unnecessary delay. Instead, the outreach of the broadcast messages can be restricted to a maximum number of hops (limited time-to-live). This restriction is not affecting the performance of the protocols, while it is significantly reducing the communication cost, as shown in figure 6.

Algorithm 4: KM Phase 1 - The information about all facilities is disseminated in the network. The user nodes and the closed facilities build a summarised view about the available open facilities in the network.

forall *open and closed facilities* **do**
    Send a facility broadcast message $m =< FID, distance = 1, status >$ to all neighbours

---

At event: receiving a facility broadcast message at user node:
if *the m.FID is not in the user local table* **then**
    Add the message payload <FID, distance, status> to the user local table record
    m.distance++
    Forward the message m to all neighbours except the source
else if *the m.distance < the current FID.distance* **then**
    Update the local table
    m.distance++
    Forward the message to all neighbours except the source
else
    Drop the message /* The same message has been received from a longer path*/

At event: receiving a broadcast message at an open facility node
m.distance++
Forward the message to all neighbours except the source

At event: receiving a broadcast message at a closed facility node:
if *m.FID is not in the local facility record* **then**
    Add the message payload <FID, distance, status> to the user local table record
    m.distance++
    Forward the message to all neighbours except the source
else if *the m.distance < the current FID.distance* **then**
    Update the local table
    m.distance++
    Forward the message to all neighbours except the source
else
    Drop the message /* The same message has been received from a longer path*/

Algorithm 5: KM Phase 2 - Find the best f_i to swap

Initialise The new medoid of the cluster = null
if *This is an open facility* **then**
    forall *Users in the cluster* **do**
        The cost of the cluster += user.cost
        forall *Closed facilities in the cluster* **do**
            The cost of the f_i += $user.f_i.cost$
    forall *Closed facilities in the cluster* **do**
        if *current cluster cost* < $f_i.cost$ **then**
            The new medoid of the cluster is f_i
    if *The new medoid of the cluster == null* **then**
        There is no swap

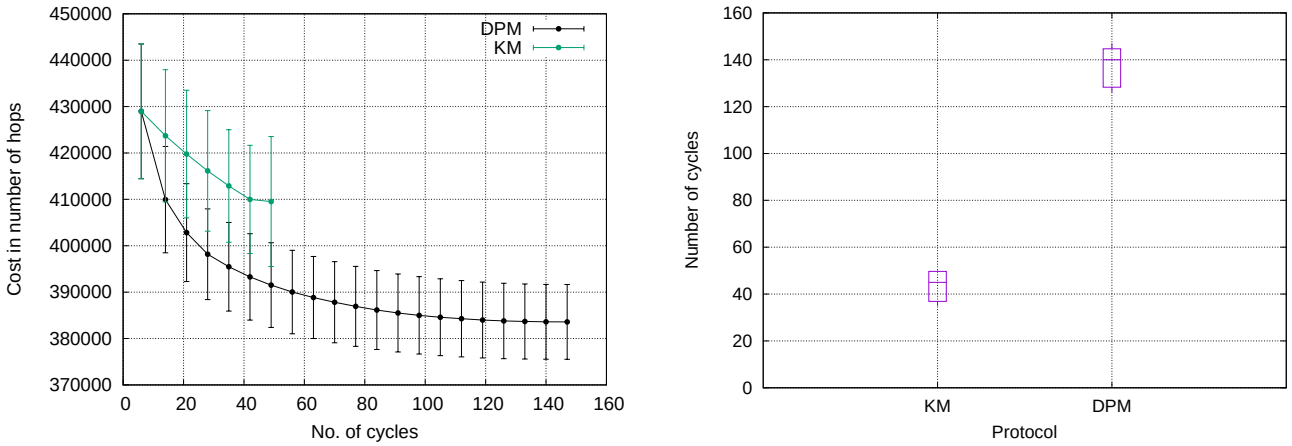
Algorithm 6: KM Phase 3 - Update the solution, the current medoid is closed and the determined f_i is opened. All the user nodes in the cluster are informed about the swap as well as all the other medoids in the solution.

```
1  if This is facility == fr then
2      facility.status = closed
3      send CHANGE-STATUS message to the fi
4      send SWAP <f_i,f_r> message to the join users and the closed facilities to inform them about the close
         decision
5      send CLUSTER-UPDATE<f_i,f_r> message to the other open facilities in the solution

6  At event: receiving a swap<f_i,f_r> message
7  update the user local table
8  determine the closest medoid
9  join the new closest medoid

10 At event: receiving a change-status message
11 facility.status = open

12 At event: receiving facility cluster-update
13 update the facility local table
14 inform the user nodes about the update
```



(a) (b)

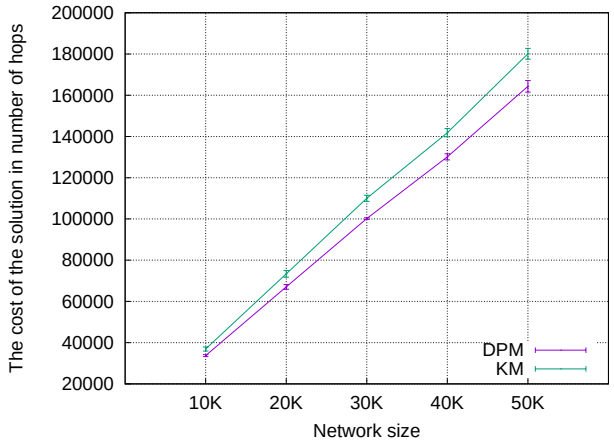
Figure 4: Convergence analysis: mean and standard deviation over 10 trials (N= 100K, m=100, p=25)

6 Conclusions

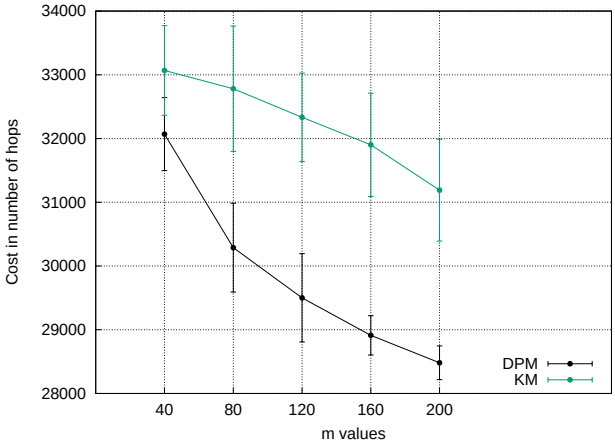
This paper has presented two novel distributed approaches for the p-median problem in networks. The proposed methods are executed without a prior knowledge of the network topology. Network information, such as the topology and the set of candidate facilities and users, do not need to be gathered for a centralised execution of a sequential algorithm. The computation in the proposed protocols is intrinsically distributed in the network nodes.

A comparative analysis over many simulations has revealed that addressing the location problem for facilities based on a global view of the network (as in DPM) leads to more accurate optimisation of the cost than clustering the network and optimising each facility location based on the local view of each cluster separately (as in KM). However, the former solution takes more cycles to converge.

The simulations have also shown that when more candidate facilities are available, both protocols can find a better solution in terms of cost. Also in this case DPM has confirmed to have better performance in taking advantage of more locations. However, when more locations are available both protocols require additional time to reach convergence.

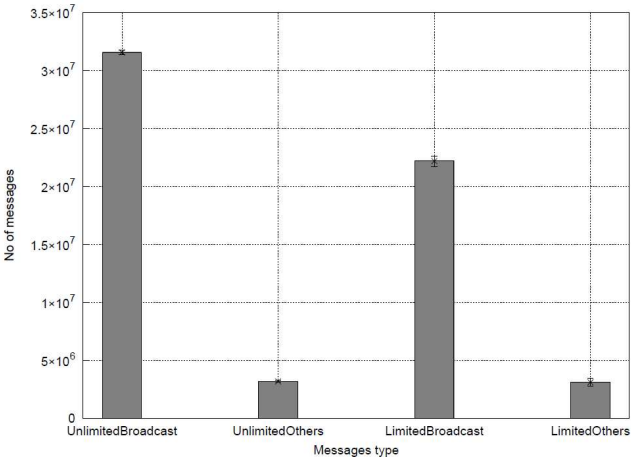


(a) Different network size

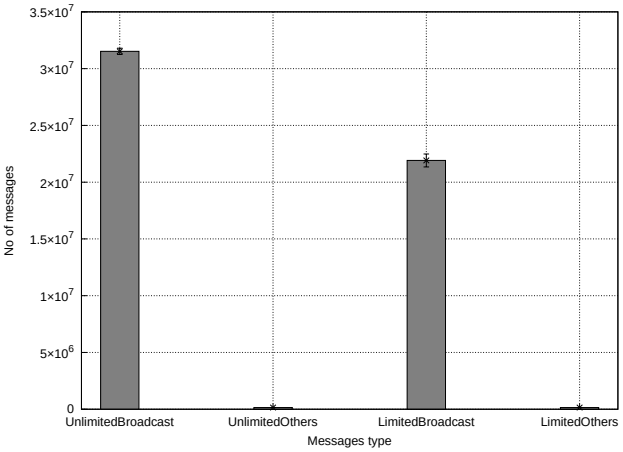


(b) Different number of candidate facilities m ($p = 25$)

Figure 5: Cost (mean and standard deviation) of the solution at convergence



(a) DPM protocol



(b) KM protocol

Figure 6: Comparison of communication overhead for broadcast and other types of messages, and for unlimited and limited (maximum time-to-live) message propagation (N= 50K, m=100, p=25)

References

[1] J. Alcaraz, M. Landete, and J. F. Monge. "Design and analysis of hybrid metaheuristics for the reliability p-median problem". In: *European Journal of Operational Research* 222.1 (2012), pp. 54–64.

[2] M. T. Melo, S. Nickel, and F. Saldanha-Da-Gama. "Facility location and supply chain management–A review". In: *European journal of operational research* 196.2 (2009), pp. 401–412.

[3] M. G. Resende and R. F. Werneck. "A fast swap-based local search procedure for location problems". In: *Annals of Operations Research* 150.1 (2007), pp. 205–230.

[4] M. B. Teitz and P. Bart. "Heuristic methods for estimating the generalized vertex median of a weighted graph". In: *Operations research* 16.5 (1968), pp. 955–961.

[5] T. S. Hale and C. R. Moberg. "Location science research: a review". In: *Annals of operations research* 123.1-4 (2003), pp. 21–35.

[6] H. Mashayekhi, J. Habibi, T. Khalafbeigi, S. Voulgaris, and M. Van Steen. "GDCluster: a general decentralized clustering algorithm". In: *IEEE transactions on knowledge and data engineering* 27.7 (2015), pp. 1892–1905.

[7] G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino. "Epidemic K-Means Clustering". In: *2011 IEEE 11th International Conference on Data Mining Workshops*. 2011, pp. 151–158.

[8] G. Di Fatta, F. Blasa, S. Cafiero, and G. Fortino. "Fault tolerant decentralised K-Means clustering for asynchronous large-scale networks". In: *Journal of Parallel and Distributed Computing* 73.3 (2013), pp. 317–329.

[9] O. Alp, E. Erkut, and Z. Drezner. "An efficient genetic algorithm for the p-median problem". In: *Annals of Operations research* 122.1-4 (2003), pp. 21–42.

[10] M. Labbé, D. Ponce, and J. Puerto. "A comparative study of formulations and solution methods for the discrete ordered p-median problem". In: *Computers & Operations Research* 78 (2017), pp. 230–242.

[11] M. Karatas, N. Razi, and H. Tozan. "A Comparison of p-median and Maximal Coverage Location Models with Q–coverage Requirement". In: *Procedia Engineering* 149 (2016), pp. 169–176.

[12] V. Marianov and D. Serra. "Median problems in networks". In: *Foundations of location analysis*. Springer, 2011, pp. 39–59.

[13] A. I. Mahmutogullari and B. Y. Kara. "Hub location under competition". In: *European Journal of Operational Research* 250.1 (2016), pp. 214–225.

[14] M. J. Hodgson, F. Shmulevitz, and M. Körkel. "Aggregation error effects on the discrete-space p-median model: The case of Edmonton, Canada". In: *Canadian Geographer/Le Géographe canadien* 41.4 (1997), pp. 415–428.

[15] R. Whitaker. "A fast algorithm for the greedy interchange for large-scale clustering and median location problems". In: *INFOR: Information Systems and Operational Research* 21.2 (1983), pp. 95–108.

[16] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno-Pérez. "The p-median problem: A survey of metaheuristic approaches". In: *European Journal of Operational Research* 179.3 (2007), pp. 927–939.

[17] L. Kaufman and P. Rousseeuw. "Clustering by means of Medoids". In: *Statistical Data Analysis Based on the L_1Norm and Related Methods*. 1987, pp. 405–416.

[18] A. Montresor and M. Jelasity. "PeerSim: A scalable P2P simulator". In: *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. IEEE. 2009, pp. 99–100.